Հայաստանի Ազգային Պոլիտեխնիկական Համալսարան

Տեղեկատվության անվտանգության և ծրագրային ապահովման ամբիոն

Արմենակ Պալյան

Ծրագրավորման ոճեր և մեթոդներ

Կուրսային նախագծի
մեթոդական ցուցումներ

ԵՐԵՎԱՆ
2018

1. Ներածություն

Կուրսային նախագիծը(ԿՆ) պատկանում է ծրագրավորման արագ զարգացող ճուղին - զուգահեռ ծրագրավորման: Զուգահեռ ծրագրի առանձին մասերը կարող են կատարվել միաժամանակ մեկ պրոցեսորում, բազմապրոցեսորային համակարգում կամ քոմփյութերային ցանցում: Այս հանգամանքը բարձրացնում է ծրագրի արագությունը: Զուգահեռ ծրագրի օրինակ կարող է հանդիսանալ **բազմահոսքային** ծրագիր: Ծրագրում (**պրոցեսում**) կարող են ստեղծվել մի քանի **հոսքեր**, որոնք կատարվում են զուգահեռաբար, սինխրոնիզացվում են իրար հետ, կատարում են փոխադարձ արգելափակում նույն **ռեսուրսի** հետ աշխատելիս: Հոսքերը օգտագործում են ռեսուրսներ, որոնք պատկանում են ամբողջ պրոցեսին: Զուգահեռ ծրագիրը կարող է նաև իրականացվել մի քանի պրոցեսներով, որոնք կատարվում են միաժամանակ:

2. Խնդրի դրվագքը

Կորպորատիվ բազմամակարդակ քոմփյութերային ցանցերում հաճախ առաջանում է հետևյալ խնդիր: Վերին մակարդակից (տնօրենի քոմփյութեր) ուղարկվում է հրահանգ ստորին մակարդակներին(կատարողների քոմփյութերներ): Ի պատասխան ստորին մակարդակներից ուղարկվում են հաղորդագրություններ, որոնք հաստատում են հրահանգի ստացումը: Մեծ կորպորացիաներում այս հաղորդագրությունների քանակը կարող է հասնել հազարների: Հաղորդագրությունները ընդունվում են տնօրենի քոմփյութերում և պետք է մշակվեն:

Եթե հաղորդագրությունները մշակվեն առանձին, դա կբերի քոմփյութերի ոչ արդյունավետ աշխատանքի: Ավելի բարձր արտադրողականությանը կարելի է հասնել, կիրառելով **փաթեթային** մշակում: Այս դեպքում հաղորդագրությունները խմբավորվում են փաթեթների մեջ և մշակվում են միասին, կտրուկ նվազեցնելուվ մեկ հաղորդագրության մշակման ժամանակը:

Հաշվողական պրոցեսը կազմակերպվում է հետևյալ ձևվով: Մի ծրագիրը ներմուծում է ցանցից եկաց հաղորդագրությունները և գրում ֆայլի մեջ, մյուս ծրագիրը կարդում է հաղորդագրությունները ֆայլից խմբերով, ստեղծելով փաթեթներ, որոնք ուղարկվում են մշակման: Երկու ծրագրերը աշխատում են զուգահեռաբար: Ֆայլի մեջ հաղորդագրությունները կազմակերպվում են **հերթի (FIFO queue)** սկզբունքով:

Ուսանողի կողմից մշակվող ծրագրերը պետք է մոդելավորեն այս պրոցեսը: Ծրագրերը պետք է ընդգրկեն տարբեր հոսքեր կամ պրոցեսներ նշվաց գործողությունները զուգահեռաբար կատարելու համար:

3. Կուրսային նախագիծ. Առաջադրանք 1

Բազմահոսքային ծրագիր DLL - ով

3.1 Ներածություն

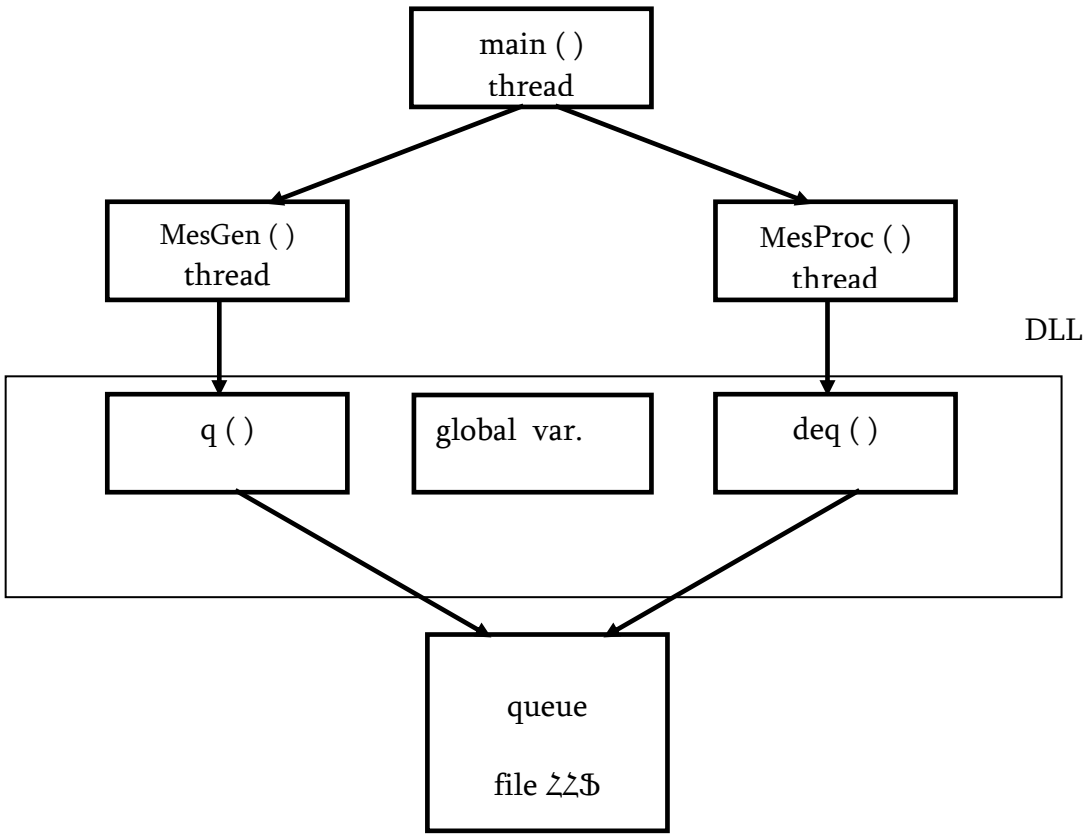
Ծրագիրը իրականացվում է **երկու** տարբերակով.
1. Համակարգային գործողությունները (հոսքերի ստեղծում և սինխրոնիզացում) իրականացվում են **WinAPI կանչերի** միջոցով։
2. Նույն գործողությունները իրականացվում են **C++11-ի դասերի** միջոցով։

Ծրագիրը ձևավորվում է **Solution**-ի մեջ 2 պրոեկտով։ **Project1** նեռարում է **prog1.cpp** ֆայլ, որը ընդգրկում է հետևյալ ֆունկցիաներ.
main ( ) – գլխավոր **հոսքի** ֆունկցիա
MesGen ( ) - հաղորդագրությունների գենեռացման **հոսքի** ֆունկցիա
MesProc ( ) - հաղորդագրությունների մշակման **հոսքի** ֆունկցիա

Project2 նեռարում է **lib.cpp** ֆայլ, որը իրականացվում է որպես **DLL** և ընդգրկում է գլոբալ փոփոխականներ (ծրագրի պարամետրներ) և հետևյալ ֆունկցիաներ.
q ( ) - ֆունկցիա, որը գրում է հաղորդագրություններ հերթի ֆայլի մեջ
deq ( ) - ֆունկցիա, որը կարդում է հաղորդագրություններ հերթի ֆայլից

Project1-ում պետք է տեղադրվի **Reference Project2** –ի վրա։

Ծրագրի ներքին կառուցվածքը.



3.2 Բազմահոսքային ծրագիր (Project1)

main ( ) գլխավոր **հոսքի** ֆունկցիան կատարում է նախապատրաստական գործողություններ, ստեղծում է 2 հոսք և սպասում նրանց ավարտի:

MesGen ( ) մոդելավորում է հաղորդագրությունների մունքը քոմփյութեր: Ֆունկցիան գենեռացնում է **tPack** հաստատուն պարբերությամբ (ծրագրի պարամետր) հաղորդագրությունների **փաթեթներ (pack)** և գրում նրանց **հաղորդագրությունների հերթի ֆայլում (ՀՀՖ):** Հաղորդագրությունների քանակը փաթեթում **պատահական է:** Հաղորդագրությունները համարակալվում են, սկսվաց մեկից: Օրինակն, **1 – ին** փաթեթ կարող են կազմել հաղորդագրություններ 1 -8, **2 – րդ** փաթեթ - 9 – 20, **3 – րդ** փաթեթ - 21 – 26: Հոսքը ավարտում է իր աշխատանքը, երբ գենեռացված հաղորդագրությունների քանակը գերազանցում է նախապես որոշվաց **ընդհանուր քանակը mesNum** (ծրագրի պարամետր):

MesProc ( ) մոդելավորում է հաղորդագրությունների մշակումը քոմփյութերում: Ֆունկցիան կարդում է ՀՀՖ – ից բոլոր գրվաց հաղորդագրությունները մեկ **խմբով (group),** դատարկելով ՀՀՖ**:** Խումբը կարող է բաղկացած լինել մի քանի MesGen – ի փաթեթներից: Այնուհետև մոդելավորվում է խմբի մշակումը, որից հետո ՀՀՖ – ից կարդացվում է նոր խումբ: Հոսքը ավարտում է իր աշխատանքը, երբ մշակված հաղորդագրությունների քանակը գերազանցում է նախապես որոշվաց **ընդհանուր քանակը mesNum** :

Ծրագրում օգտագործվում են **Windows API կանչերը (C++11-ի դասերը)** հետևյալ նպատակով.

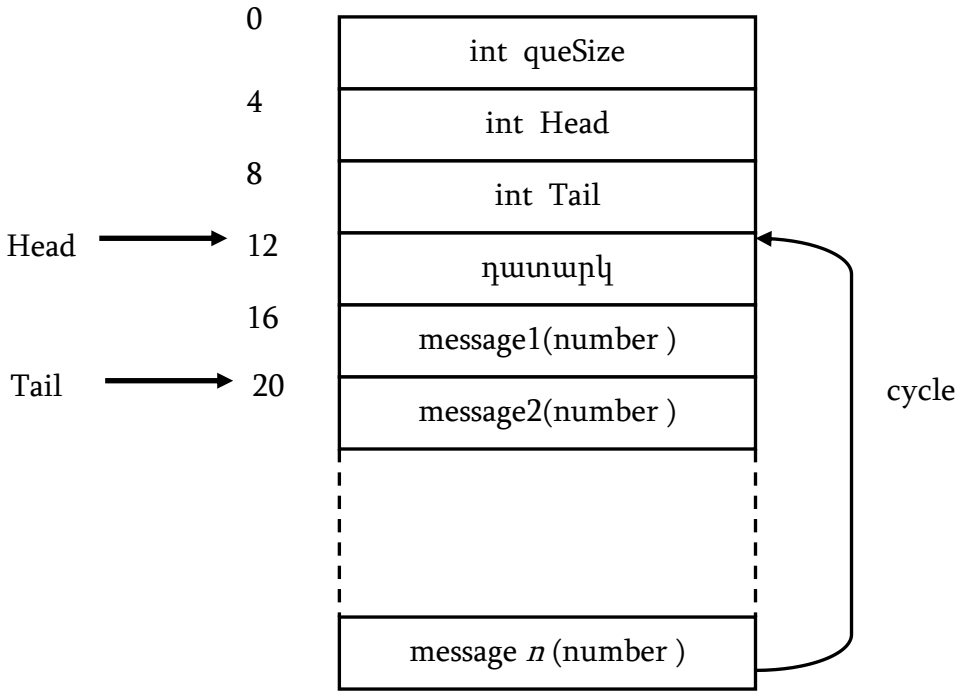
1. Հոսքերի ստեղծում
2. ՀՀՖ – ի հետ երկու հոսքերի միաժամանակ աշխատանքի փոխադարձ արգելափակում **critical section – ի** (C++11-ի **mutex-ի**) միջոցով

Ծրագիրը արտածում է հաղորդագրություններ էկրանի վրա և հատուկ տեքստային **հաշվառման ֆայլի** մեջ հատագա վերլուծության համար:

3. 3. Հաղորդագրությունների հերթի ֆայլ (ՀՀՖ)

ՀՀՖ –ն - **բինար ֆայլ** է: Ֆայլի մեջ հաղորդագրությունները կազմակերպվում են **ցիկլիկ հերթի (FIFO queue)** սկզբունքով: ՀՀՖ – ի մեջ հաղորդագրությունը գրվում է իր համարով (*int* արժեք):

ՀՀՖ ունի հետևյալ կառուցվածք.



ՀՀՖ –ն ունի վերնագիր, որը ներկայացվում է ստրուկտուրայով.

```
struct   queHeader {
            int   queSize;              //number of entries in queue
                int    Head;               //offset in file
                int    Tail;               //offset in file
              };
```

queSize – հաղորդագրությունների (հերթի տարրերի) մաքսիմալ քանակը ՀՀՖ – ում
Head - ցուցիչ (շեղում ֆայլում) այն դատարկ տարրի վրա, որին հաջորդում է առաջին հերթից հանվող տարրը – **հերթի սկիզբ** (message1)
Tail - ցուցիչ (շեղում ֆայլում) հերթի վերջին տարրի վրա – **հերթի վերջ** (message2)

Շեղումը օգտագործվում է fseek ( ) ֆունկցիայում: Ֆայլի հետ "գրել – կարդալ" գործողությունները կատարվում են fwrite ( ), fread ( ) ֆունկցիաներով:

Head և Tail – ի սկզբնական արժեքները - 12
Հերթից 1 տարր հանելուց առաջ կատարվում է.
Head = Head + 4;
Հերթի մեջ 1 տարր տեղադրելուց առաջ կատարվում է.
Tail = Tail + 4;

Հերթը աշխատում է **ցիկլիկ** սկզբունքով: **Դա նշանակում է, որ Head և Tail –ը, հասնելով հերթի վերջը, ստանում են սկզբնական արժեք - 12:**

Հերթի **դատարկ** լինելու պայմանը՝ Head == Tail: Նույն պայմանը առաջանում է , երբ հերթը լինում է **լցված** վիճակում:

ՀՀՖ – ի մաքսիմալ չափը որոշվում է.

```
int  totSize = queSize * 4+12;
```

ՀՀՖ –ն բացելուց հետո նրա մեջ գրվում է վերնագրի **սկզբնական արժեքը**.

```
queHeader  qHead = {100,12,12};
```

Դա նշանակում է՝ queSize = 100

totSize, qHead - տեղադրվում են ծրագրի գլոբալ տիրույթում

Վերնագրի **ընթացիկ արժեքը** գտնվում է ֆայլի մեջ և օգտագործվում է ծրագրի կատարման ժամանակ:

ՀՀՖ – ի մեջ հաղորդագրությունները տեղադրվում են MesGen ( ) հոսքի կողմից, իսկ հանվում են MesProc ( ) հոսքի կողմից: Եթե MesProc ( ) հոսքը հասցնում է հանել հաղորդագրությունները ՀՀՖ – ից, ապա ՀՀՖ – ի **գերլցում** չի առաջանում: Հոսքերը գտնվում են **բալանսի** մեջ: Հակառակ դեպքում առաջանում է ՀՀՖ – ի **գերլցում,** որի հետևանքով կատարվում է ծրագրի վթարային ավարտ (exit (1)):

ՀՀՖ – ի **գերլցման** հավանականությունը ուղղակի կախված է **խմբի** չափի միջին արժեքից **(Avr): (qHead. queSize – Avr)** տարբերության նվազումը բերում է հավանականության բարձրացման:

Հոսքերի **բալանսը** որոշվում է ծրագրի պարամետրների ընտրությամբ:

3. 4. Ծրագրի հիմնական պարամետրները

Ստորև ներկայացվում է հիմնական պարամետրները.

```
const  int  maxPackSize = 20;        //max packet size
const  int  mesNum = 1000;           //total number of  messages
const  int  tPack = 100;             //period of packets (msec)
const  float  mesPr = 1.0;           //coefficient of message processing time

const  queHeader qHead = {100,12,12};    // initial parameters of  queue
const  int totSize = qHead.queSize * 4+12;   //total size of queue file
```

mesNum - գենեռացվող հաղորդագրությունների ընդհանուր քանակը
tPack - հաղորդագրությունների փաթեթների գենեռացման պարբերությունը (msec)
qHead - ՀՀՖ – ի սկզբնական վերնագիր
maxPackSize - հաղորդագրությունների փաթեթի մաքսիմալ չափը
totSize - ՀՀՖ – ի մաքսիմալ չափը:
mesPr - հաղորդագրությունների մշակման ժամանակի գործակցի հաստատուն մասը:

Պարամետրեր tPack, qHead.queSize, maxPackSize, mesPr որոշում են MesGen և MesProc հոսքերի բալանսը: Փոքրացնելով, օրինակ, tPack պարամետրը կարելի է բալանսը խախտել, ինչը կբերի ծրագրի անաշխատունակությանը: Ներկայացված արժեքները ապահովում են հոսքերի բալանսը որոշ քոմփյութերների վրա կախված նրանց

արագագործությունից: **Ուսանողը** կարող է որոշ արժեքներ փոխի **իր քոմփյութերում** բալանսը ապահովելու համար:

Ծրագրի հիմնական պարամետրները տեղադրվում են DLL – ում:

3. 5 main ( ) գլխավոր հոսքի ֆունկցիան

main ( ) գլխավոր **հոսքի** ֆունկցիան.

1. Նախապատրաստում է աշխատանքի (սկզբնավորում, բացում) հետևյալ տարրեր.
   - **critical section** – ի (**C++11-ի mutex-ի**) արգելափակող օբյեկտը
   - պատահական թվերի գեներատոր
   - ՀՀՖ բինար ֆայլ (que.dat)
   - հաշվառման տեքստային ֆայլ (log.txt)
2. Գրում է ՀՀՖ –ի մեջ վերնագրի **սկզբնական արժեքը**.
3. Արտածում և հաշվարում է **log.txt** ֆայլում հաղորդագրություն ծրագրի ընդհանուր **սկզբի** մասին:
4. Գործարկում է MesGen ( ) և MesProc ( ) հոսքերի ֆունկցիաները
5. Կատարում է նշվաց հոսքերի սպասում
6. Փակում է **արգելափակող օբյեկտը** և ֆայլերը
7. Արտածում և հաշվարում է **log.txt** ֆայլում հաղորդագրություն ծրագրի ընդհանուր **ավարտի** մասին:

3. 6 MesGen ( ) - հաղորդագրությունների գենեռացման հոսքի ֆունկցիա

MesGen ( ) մոդելավորում է հաղորդագրությունների մունքը քոմփյութեր: Ֆունկցիան գենեռացնում է **tPack** հաստատուն պարբերությամբ հաղորդագրությունների **փաթեթներ (pack)** և գրում նրանց **հաղորդագրությունների հերթի ֆայլում (ՀՀՖ):** Հաղորդագրությունների քանակը փաթեթում **պատահական է:** Հաղորդագրությունները համարակալվում են, սկսվաց մեկից: Օրինակն, **1 – ին** փաթեթ կարող են կազմել հաղորդագրություններ 1 - 8, **2 – րդ** փաթեթ - 9 – 20, **3 – րդ** փաթեթ - 21 – 26: Հոսքը ավարտում է իր աշխատանքը, երբ գենեռացված հաղորդագրությունների քանակը գերազանցում է նախապես որոշվաց **ընդհանուր քանակը mesNum** (ծրագրի պարամետր):Արդյունքում գենեռացվում են հաղորդագրություններ **1 - mesNum** համարներով:

Ստորև ներկայացվում է ֆունկցիայի **պսեվդոկոդը.**

```
.....MesGen(....)     //message generation thread
{
       int pack,count = 0,mes [100]= {0};

  //count – total number of generated messages
  while( count< mesNum )
  {
    pack = պատահական թիվ (1 – maxPackSize) տիրույթում;
    Գրել mes [100] – ի մեջ գեներացված հաղորդագրությունների հաջորդական համարները;
    count+= pack;
    Գրավել արգելափակող օբյեկտը ;
    q(mes,pack,fp);                    //place packet in queue
                                       // fp - բացված ՀՀՖ-ի ցուցիչ
    cout<<"Place in queue "<<pack<<endl;
    fprintf( ....., "Place in queue %d\n",pack ); //place in log file

    Ազատել արգելափակող օբյեկտը ;


    Sleep(tPack);                  //pause for the next packet.

 }//end of while

   cout<< "End of message generation"<< endl;
   fprintf( ....., "End of message generation\n"); //place in log file

   return 0;
}
```

3. 7 MesProc ( ) - հաղորդագրությունների մշակման հոսքի ֆունկցիա

MesProc ( ) մոդելավորում է հաղորդագրությունների մշակումը քոմփյութերում: Ֆունկցիան կարդում է ՀՀՖ – ից **բոլոր** գրվաց հաղորդագրությունները մեկ **խմբով (group):** Խումբը կարող է բաղկացած լինել **մի քանի MesGen – ի փաթեթներից**: Այնուհետև մոդելավորվում է խմբի մշակումը, որից հետո ՀՀՖ – ից կարդացվում է նոր խումբ: Հոսքը ավարտում է իր աշխատանքը, երբ մշակված հաղորդագրությունների քանակը գերազանցում է նախապես որոշված **ընդհանուր քանակը mesNum** :

MesProc ( ) մոդելավորում է խմբի մշակումը հետևյալ ցիկլով.

```
float time =( mesPr + group/100)* 1e8;
for ( i = 0; i < time; ++ i );
```

որտեղ **mesPr** - հաղորդագրությունների մշակման ժամանակի գործակցի հաստատուն , իսկ **group/100** - փոփոխական մասերն են:

MesProc ( ) հաշվառում է բոլոր ստացված հաղորդագրությունները իրենց համարներով: Նա պարունակում է **char flag[1200]** զանգված,որտեղ **flag[k - 1]** –ին վերագրվում է **'1'** հաղորդագրություն **k** համարով ստանալիս: **MesProc ( ),** ավարտելով իր աշխատանքը, ստուգում է, որ բոլոր 1 – **mesNum** հաղորդագրությունները ստացված են:

Ստորև ներկայացվում է ֆունկցիայի **պսեվդոկոդը.**

```
.....MesProc(....)     //message processing thread
{
      int group,count=0,mes [200]= {0};
     int sum=0,cnt=0;
     char flag[1200] = {0};

  while( count< mesNum )
 {

     Գրավել արգելափակող օբյեկտը ;
     group= deq(mes,fp);   //take message group from queue
                           // fp - բացված ՀՀՖ-ի ցուցիչ
     cout<<"Take from  queue "<< group<<endl;
     fprintf( ....., "Take from queue %d\n", group); //place in log file

     Ազատել արգելափակող օբյեկտը ;

    sum+=group;    //խմբի չափի միջին  արժեք (Avr)հաշվելու համար
     cnt++;

    if( group == 0)
     {
      Sleep(200);
     }
    else
     {
       count+=group;
       float time =( mesPr + group/100)* 1e8;
       for ( i = 0; i < time; ++ i );

       Հաշվառել խմբի բոլոր  հաղորդագրությունները;

    }//end of else
 }//end of while

    Ստուգել, որ բոլոր 1 – mesNum հաղորդագրությունները  ստացված են:
    Հաշվել և արտածել  խմբի չափի միջին  արժեքը (float Avr):
    cout << "End of message processing"<<endl;
    fprintf( ....., "End of message processing\n");//place in log file
return 0;
}
```

3.8 DLL (Project2)

DLL բաղկացած է 2 մասից.

1. **Գլոբալ փոփոխականների** տիրույթ, որտեղ տեղադրվում են ծրագրի պարամետրները (mesNum, tPack և այլն):
2. **Ֆունկցիաներ** q ( ) և deq ( ):

Գլոբալ փոփոխականները և ֆունկցիաները պետք է **հասանելի լինեն prog1.cpp (Project1)** – ին: Նշված անունները պետք է **export** արվեն DLL-ից և ունենան **արտաքին շաղկապում (external linkage)** : Անունների հասանելիությունը **prog1.cpp**-ից պետք է ապահովվի սիմետրիկ գործողություններով – **import** և այլն:

DLL –ի սկզբնական կոդը ձևավորվում է **lib.cpp** ֆայլի տեսքով, որը ընդգրկվում է **Project2** - ի մեջ:

q ( ) ֆունկցիա

q ( ) - ֆունկցիա, որը գրում է հաղորդագրությունների փաթեթ ՀՀՖ – ի մեջ

Ֆունկցիան ունի պրոտոտիպ.

int q(int arr[ ],int size,FILE *fp);

Պարամետրներ.

int arr[ ] - զանգվածի հասցե, որում գրված են հաղորդագրությունների համարները
int size - հաղորդագրությունների քանակը
FILE * fp - բացված ՀՀՖ – ի ցուցիչ

Գործողություններ.

Կարդալ ՀՀՖ –ից վերնագրի ընթացիկ արժեքը
Ցիկլով գրել ՀՀՖ – ի մեջ մեկը մյուսի հետևից փոխանցված հաղորդագրությունները:
Գրել ՀՀՖ –ի մեջ փոփոխվաց վերնագիրը

Ցիկլում կատարել հետևյալ **ստուգում** `

if (հերթական հաղորդագրություն գրելիս կառաջանա ՀՀՖ – ի **գերլցում)**
return 1; //error

Նորմալ ավարտի դեպքում - **return 0;**

deq ( ) ֆունկցիա

deq ( ) - ֆունկցիա, որը կարդում է հաղորդագրությունների խումբ ՀՀՖ – ից

Ֆունկցիան ունի պրոտոտիպ.

```
int  deq( int arr[ ], FILE *fp );
```

Պարամետրներ.

int arr[ ] - զանգվածի հասցե, որի մեջ գրվում են կարդացված հաղորդագրությունների համարները

FILE * fp - բացված ՀՀՖ – ի ցուցիչ

Գործողություններ.

Կարդալ ՀՀՖ –ից վերնագրի ընթացիկ արժեքը

Ցիկլով կարդալ ՀՀՖ – ից մեկը մյուսի հետևից հաղորդագրություններ մինչև **հերթի դատարկվելը**:

Գրել ՀՀՖ –ի մեջ փոփոխված վերնագիրը

return k; // կարդացված հաղորդագրությունների քանակը խմբում

Ծանոթագրություն ՀՀՖ –ի մեջ **գրելու** (**fwrite()**) գործողությունից հետո 2 ֆունկցիաներում կիրառել **fflush ( )** ֆունկցիան:

4. Կուրսային նախագիծ. Առաջադրանք 2

Բազմապրոցեսային ծրագիր DLL - ով

4.1 Ներածություն

Ծրագիրն իրականացնում է նույն խնդիրը 2 զուգահեռ **պրոցեսների** միջոցով՝ պրոցեսներ MesGen և MesProc: **MesGen** պրոցեսը մոդելավորում է հաղորդագրությունների մունքը քոմփյութեր: **MesProc** պրոցեսը մոդելավորում է հաղորդագրությունների մշակումը քոմփյութերում: Սկզբից գործարկվում է **MesGen –ը,** որը մի քանի նախապատրաստական գործողություններ կատարելուց հետո գործարկում է **MesProc –ը:** Այս պահից հետո 2 պրոցեսները կատարվում են **զուգահեռ տարբերակ 1-ի նման**: Ամեն մի պրոցեսի շրջանակում կատարվում է առաձին ծրագիր (.cpp և .exe ֆայլեր):

Ծրագիրը ձևավորվում է **Solution**-ի մեջ 3 պրոեկտով.

Project1 ներառում է **prog2.cpp** ֆայլ, որը իրականացնում է **MesGen** պրոցեսը համապատասխան main ( ) ֆունկցիայով:
Project3 ներառում է **prog3.cpp** ֆայլ, որը իրականացնում է **MesProc** պրոցեսը համապատասխան main ( ) ֆունկցիայով:

Project2 ներառում է **lib.cpp** ֆայլ, որը իրականացվում է որպես **DLL** և ընդգրկում է գլոբալ փոփոխականներ (ծրագրի պարամետրներ) և հետևյալ ֆունկցիաներ.
q ( ) - ֆունկցիա, որը գրում է հաղորդագրություններ հերթի ֆայլի մեջ
deq ( ) - ֆունկցիա, որը կարդում է հաղորդագրություններ հերթի ֆայլից

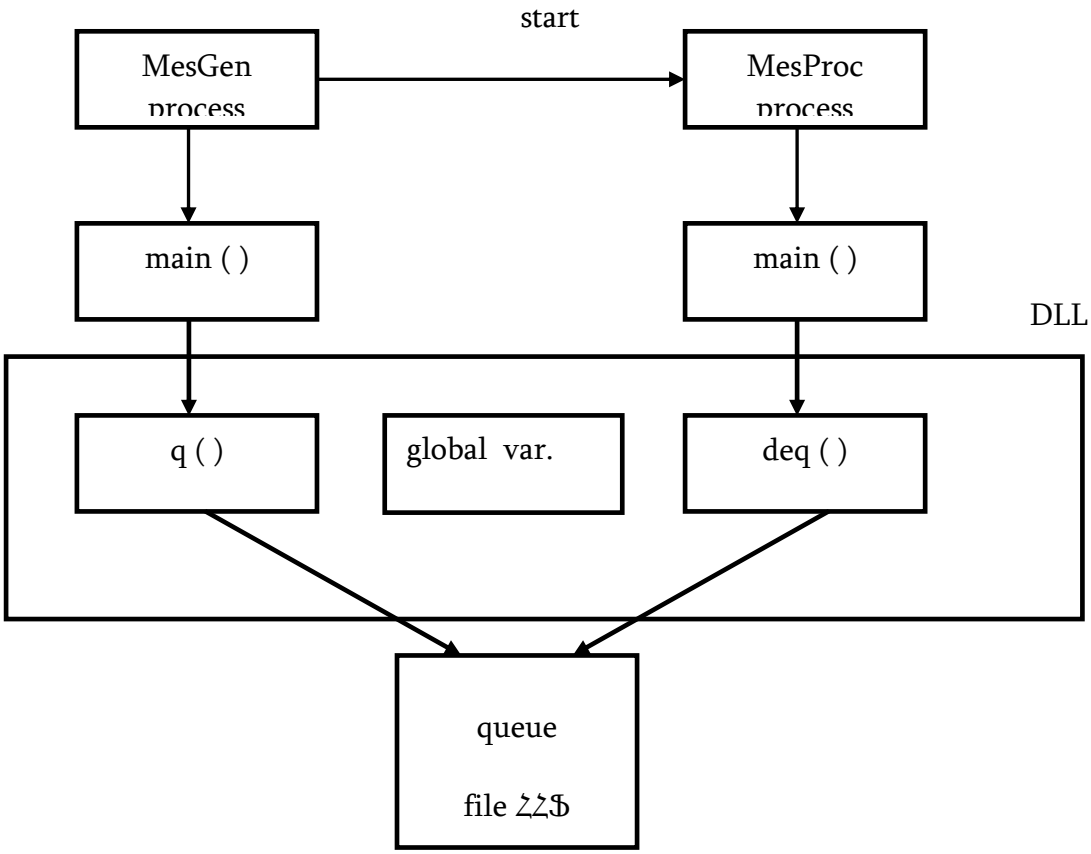
DLL Առաջադրանքներ 1,2-ում նույն է:

DLL-ի անունների հասանելիությունը **prog2.cpp, prog3.cpp** -ից պետք է ապահովվի սիմետրիկ գործողություններով (**import** և այլն) **Առաջադրանք 1-ի** նման:

Project1-ում և 3-ում պետք է տեղադրվի **Reference Project2** –ի վրա:

4. 2 Բազմապրոցեսային ծրագիր

Ծրագրի ներքին կառուցվածքը.



ՀՀՖ – ի հետ երկու պրոցեսների միաժամանակ աշխատանքի փոխադարձ արգելափակում կատարել **Windows API mutex** – ի միջոցով: Պրոցեսները արտածում են հաղորդագրություններ էկրանի վրա մեկ պատուհանի (console -ի) մեջ: Հատուկ **հաշվառման ֆայլ** այս տարբերակում չի ստեղծվում:

Ստորև ներկայացվում է MesGen պրոցեսի main( ) ֆունկցիայի **պսեվդոկոդը.**

```
.....main(....)      //MesGen process
{
```

Նախապատրաստել աշխատանքի (սկզբնավորել, բացել) հետևյալ տարրեր.

- պատահական թվերի գեներատոր
- ՀՀՖ բինար ֆայլ (que.dat)
- **mutex** – ի օբյեկտը

Գրել ՀՀՖ –ի մեջ վերնագրի **սկզբնական արժեքը**.

Գործարկել MesProc պրոցեսը **CreateProcess()** API-կանչով (նույն console - ով)

ՀՀՖ –ի մեջ **փաթեթների** գրման ցիկլ տարբերակ 1-ի նման այն տարբերությամբ, որ կատարվում է **Windows API mutex** – ի գրավում (ազատում):

Սպասել MesProc պրոցեսի ավարտը

Փակել դեսկրիպտորներ.

- **mutex** – ի
- MesProc պրոցեսի և նրա հոսքի

Փակել ՀՀՖ ֆայլը

Արտածել հաղորդագրություն ավարտի մասին

}

ՀՀՖ –ի գերլցման **դեպքում** պետք է դադարեցվի MesProc պրոցեսի աշխատանքը **TerminateProcess** API-կանչով սեփական պրոցեսի վթարային ավարտ կատարելուց առաջ:

Ստորև ներկայացվում է MesProc պրոցեսի main( ) ֆունկցիայի **պսեվդոկոդը.**

.....main(....) //MesProc process

{

Բացել ՀՀՖ ֆայլը (որպես գոյություն ունեցող ֆայլ)

Բացել գոյություն ունեցող **mutex** –ի օբյեկտը

ՀՀՖ –ից խմբերի կարդալու ցիկլ տարբերակ 1-ի նման այն տարբերությամբ, որ կատարվում է **Windows API mutex** – ի գրավում (ազատում):

Ստուգել, որ բոլոր 1 – mesNum հաղորդագրությունները ստացված են:

Հաշվել և արտածել խմբի չափի միջին արժեքը (float Avr):

Փակել mutex –ի դեսկրիպտոր

Արտածել հաղորդագրություն ավարտի մասին

}

4.3. Լրացուցիչ պահանջներ

1. Քանի որ տարբեր պրոցեսներ կատարում են զուգահեռաբար մուտք-ելքի գործողություններ ՀՀՖ ֆայլի նետ, **ամեն մի գրելուց (fwrite()) հետո օգտագործել fflush()** ֆունկցիա ապահովելու համար ինֆորմացիայի իրական գրանցումը ֆայլում:

2. Windows API կանչերի և նրանց հետ կապվաց ստրուկտուրաների օգտագործման դեպքում

CreateProcess

STARTUPINFO

CreateMutex

OpenMutex

հաշվի առնել, որ այս կանչերում որպես առանձին պարամետր օգտագործվում է տող, օրինակ, **mutex** –ի անունը: Եթե նա - ավանդական տող է 1-բայտանոց կոդավորմամբ (ANSI տող), ապա VC++2015 –ում նշվաց անուններին պետք է վերջում ավելացնել **A** տառը, օրինակ, **CreateProcessA**.

UNICODE տողերի դեպքում A տառը չի ավելանում:

3. CreateProcess կանչում գործարկվող .exe ֆայլի անունը տալ **երկրորդ պարամետրում (lpCommandLine),** չնշելով լրիվ ուղին:

5. Ծրագրերի մշակման և բացատրագրի ձևակերպման պահանջները

1. Ծրագրերը մշակվում են C++ լեզվով, օգտագործելով Win API կանչերը կամ C++11-ի դասերը, Windows- ի միջավայրում աշխատելու համար: Միջավայր՝ Visual C++ 2015, Project – Console Application.

2. Ծրագրերի մշակումը պետք է կատարվի խստորեն ըստ սույն մեթոդական ցուցումների, պահպանելով նաև փոփոխականների և ֆունկցիաների անունները:

3. Բացատրագիրն ընդգրկում է՝

Առաջադրանք 1

Վերնագրի էջ

1. Զուգահեռ ծրագրերի մշակում Windows – ի միջավայրում:
   Պրոցեսի և հոսքի գաղափարները:
   Windows API կանչերը (C++11-ի դասերը), որոնք ապահովում են.
   - Հոսքերի ստեղծում և սինխրոնիզացում նրանց հետ,
   - Հոսքերի միաժամանակ աշխատանքի փոխադարձ արգելափակում **critical section** – ի և C++11-ի **mutex** – ի միջոցով:
2. Երկու տարբերակների ծրագրերի նկարագրությունը, նեռարյալ DLL :
   Ըստ տվյալ ձեռնարկի
3. Տարբերակ 1-ի prog1.cpp (Project1) տպված սկզբնական կոդը (նեռարյալ .h ֆայլերը):
4. Տարբերակ 2-ի prog1.cpp (Project1) տպված սկզբնական կոդը (նեռարյալ .h ֆայլերը):
5. lib.cpp (Project2) տպված սկզբնական կոդը (գլոբալ փոփոխականներ և ֆունկցիաներ):
6. Կամայական տարբերակի կատարման ընթացքում հաշվառվաց հաղորդագրությունները (տպված **log ֆայլը**):

Առաջադրանք 2

Վերնագրի էջ

1. Զուգահեռ ծրագրերի մշակում Windows – ի միջավայրում:
   Պրոցեսի և հոսքի գաղափարները:
   Windows API կանչերը , որոնք ապահովում են.
   - Հոսքերի և պրոցեսների ստեղծում և սինխրոնիզացում նրանց հետ,
   - Հոսքերի և պրոցեսների միաժամանակ աշխատանքի փոխադարձ արգելափակում Windows API **mutex** – ի միջոցով:
2. Ծրագրերի նկարագրությունը, նեռարյալ DLL :
   Ըստ տվյալ ձեռնարկի
3. prog2.cpp (Project1) տպված սկզբնական կոդը (նեռարյալ .h ֆայլերը):
4. prog3.cpp (Project3) տպված սկզբնական կոդը (նեռարյալ .h ֆայլերը):
5. lib.cpp (Project2) տպված սկզբնական կոդը (գլոբալ փոփոխականներ և ֆունկցիաներ):

Բացատրագրի հետ ներկայացվում է կրիչ (DVD), որը պարունակում է ծրագրերի պրոեկտները և կատարվող ֆայլերը:

6. Պահանջներ ԿՆ հանձնելու համար

Ուսանողը պարտավոր է .

1. Ցուցաբերել ծրագրերի ամբողջ կոդի իմացությունը:
2. Աշխատեցնել ծրագրերը նորմալ ռեժիմում:
3. Իմանալ և ցույց տալ գործնականորեն, ինչպես են ազդում ծրագրերի աշխատանքի վրա հոսքերի բալանսի առումով հետևյալ պարամետրերը.
   maxPackSize , tPack , mesPr , qHead.queSize
4. Բացատրել, ինչպես ծրագրերը կաշխատեն բազմապրոցեսորային քոմփյութերներում:

Գրականություն

1. Побегайло А. Системное программирование в Windows.- С.-Пб: БХВ-Петербург, 2006
2. Hart M. Windows System Programming. AW Professional. 2004.